

FULL STACK DEVELOPMENT

TASK 2

1. Build a basic HTTP server that can handle different routes and HTTP methods (GET, POST, PUT, DELETE).

PROGRAM:

```
const http = require('http');

// Define the routes and their handlers

const routes = {

  '/': {

    GET: (req, res) => {

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.end('Welcome to the Home Page');

    }

  },

  '/about': {

    GET: (req, res) => {

      res.writeHead(200, { 'Content-Type': 'text/plain' });

      res.end('About Us Page');

    }

  },

  '/api/data': {

    GET: (req, res) => {

      res.writeHead(200, { 'Content-Type': 'application/json' });

      res.end(JSON.stringify({ message: 'This is a GET request' }));

    },

    POST: (req, res) => {

      let body = '';

      req.on('data', chunk => {

        body += chunk.toString();

      });

      req.on('end', () => {

        res.writeHead(201, { 'Content-Type': 'application/json' });

        res.end(JSON.stringify({ message: 'Data received', data: body }));

      });

    },

    PUT: (req, res) => {
```

```

    let body = "";
    req.on('data', chunk => {
        body += chunk.toString();
    });
    req.on('end', () => {
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ message: 'Data updated', data: body }));
    });
},
DELETE: (req, res) => {
    res.writeHead(200, { 'Content-Type': 'application/json' });
    res.end(JSON.stringify({ message: 'Data deleted' }));
}
}
};

```

// Create the HTTP server

```

const server = http.createServer((req, res) => {
    const { method, url } = req;

    // Check if the route exists
    if (routes[url] && routes[url][method]) {
        // Call the handler for the route and method
        routes[url][method](req, res);
    } else {
        // Route not found
        res.writeHead(404, { 'Content-Type': 'text/plain' });
        res.end('404 Not Found');
    }
});

```

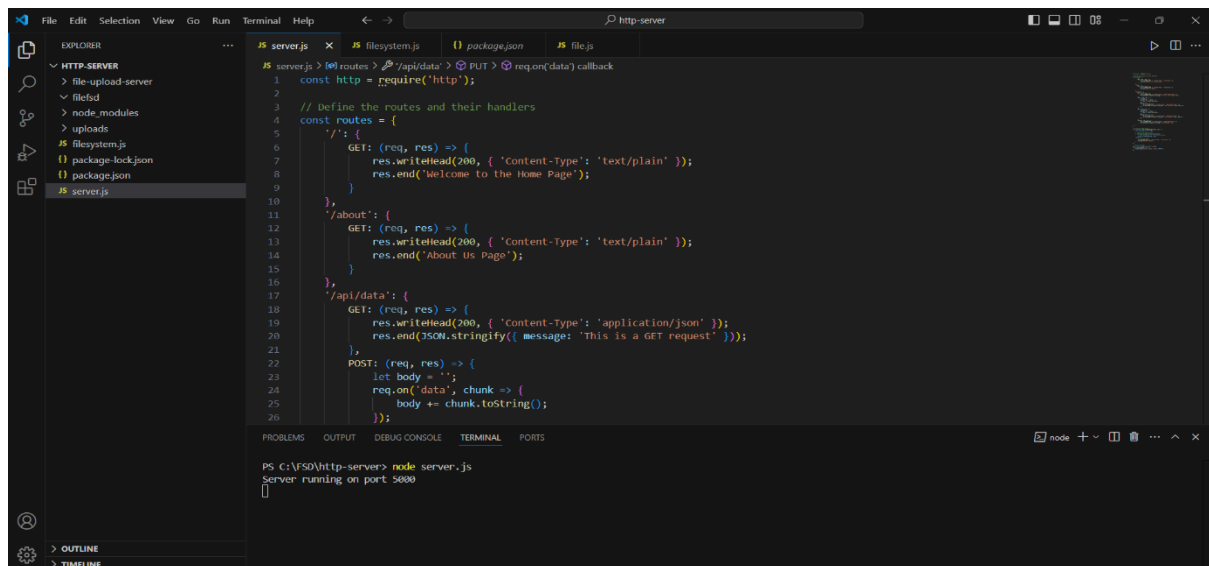
// Start the server

```

const PORT = process.env.PORT || 5000;
server.listen(PORT, () => {
    console.log(`Server running on port ${PORT}`);
});

```

OUTPUT



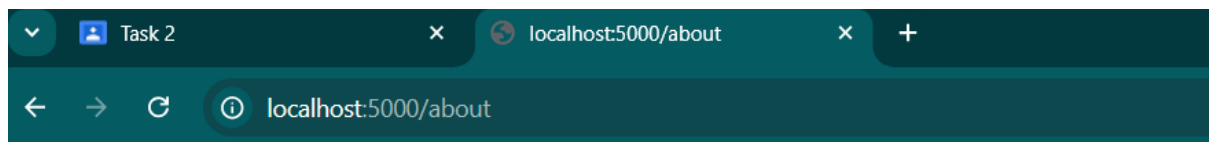
The screenshot shows the VS Code editor with the `server.js` file open. The file contains the following code:

```
1 const http = require('http');
2
3 // Define the routes and their handlers
4 const routes = {
5   '/': {
6     GET: (req, res) => {
7       res.writeHead(200, { 'Content-Type': 'text/plain' });
8       res.end('Welcome to the Home Page');
9     },
10  },
11  '/about': {
12    GET: (req, res) => {
13      res.writeHead(200, { 'Content-Type': 'text/plain' });
14      res.end('About Us Page');
15    },
16  },
17  '/api/data': {
18    GET: (req, res) => {
19      res.writeHead(200, { 'Content-Type': 'application/json' });
20      res.end(JSON.stringify({ message: 'This is a GET request' }));
21    },
22    POST: (req, res) => {
23      let body = '';
24      req.on('data', chunk => {
25        body += chunk.toString();
26      });
27    },
28  },
29 }
```

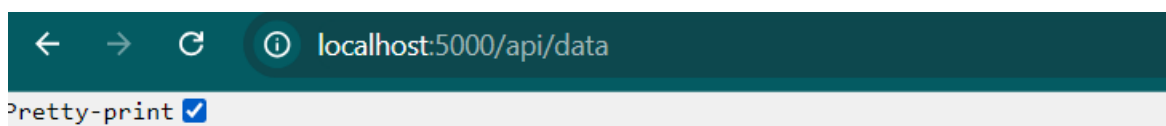
The terminal output shows the command `node server.js` being executed, and the server running on port 5000.



Welcome to the Home Page



About Us Page



```
{
  "message": "This is a GET request"
}
```

```
Command Prompt
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\vishn>curl http://localhost:5000/api/data
{"message":"This is a GET request"}
C:\Users\vishn>curl -X POST http://localhost:5000/api/data -d "name=John&age=30"
{"message":"Data received","data":{"name=John&age=30"}}
C:\Users\vishn>curl -X PUT http://localhost:5000/api/data -d "name=Jane&age=25"
{"message":"Data updated","data":{"name=Jane&age=25"}}
C:\Users\vishn>curl -X DELETE http://localhost:5000/api/data
{"message":"Data deleted"}
C:\Users\vishn>curl -X POST http://localhost:5000/api/data -d "name=John&age=30"
{"message":"Data received","data":{"name=John&age=30"}}
C:\Users\vishn>
```

2. Create a server that allows users to upload files and save them to the server's filesystem.

PROGRAM

```
const express = require('express');
const multer = require('multer');
const path = require('path');
const app = express();

// Set storage engine
const storage = multer.diskStorage({
  destination: './uploads/',
  filename: (req, file, cb) => {
    cb(null, `${Date.now()}-${file.originalname}`);
  },
});

const upload = multer({
  storage: storage,
  limits: { fileSize: 1000000 }, // Limit file size to 1MB
  fileFilter: (req, file, cb) => {
    checkFileType(file, cb);
  },
}).single('myFile'); // Accept a single file with the key name 'myFile'

// Check file type
function checkFileType(file, cb) {
```

```
const filetypes = /jpeg|jpg|png|gif/;

const extname = filetypes.test(path.extname(file.originalname).toLowerCase());

const mimetype = filetypes.test(file.mimetype);

if (mimetype && extname) {

  return cb(null, true);

} else {

  cb('Error: Images Only!');

}

}

app.use(express.static('./public'));

app.get('/', (req, res) => res.send('File Upload Server'));

app.post('/upload', (req, res) => {

  upload(req, res, (err) => {

    if (err) {

      res.status(400).send(err);

    } else {

      if (req.file === undefined) {

        res.status(400).send('Error: No File Selected!');

      } else {

        res.send(`File uploaded: ${req.file.filename}`);

      }

    }

  });

});

const PORT = process.env.PORT || 4000;

app.listen(PORT, () => console.log(`Server started on port ${PORT}`));
```

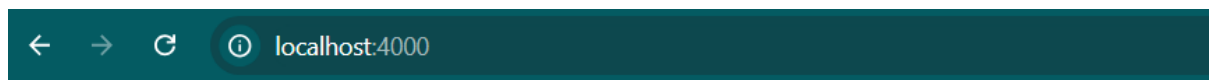
The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays the project structure for 'HTTP-SERVER', including folders like 'file-upload-server', 'node_modules', 'public', and 'uploads', and files like 'package-lock.json', 'package.json', 'server.js', and 'filesystem.js'. The main editor area shows the 'filesystem.js' file with the following code:

```
JS filesystem.js > PORT
1  const express = require('express');
2  const multer = require('multer');
3  const path = require('path');
4  const app = express();
5
6  // Set storage engine
7  const storage = multer.diskStorage({
8    destination: './uploads/',
9    filename: (req, file, cb) => {
10      cb(null, `${Date.now()}-${file.originalname}`);
11    },
12  });
13
14  // Initialize upload variable
15  const upload = multer({
16    storage: storage,
17    limits: { fileSize: 1000000 }, // Limit file size to 1MB
18    fileFilter: (req, file, cb) => {
19      checkFileType(file, cb);
20    },
21  }).single('myFile'); // Accept a single file with the key name 'myFile'
22
23  // Check file type
24  function checkFileType(file, cb) {
25    // Allowed ext
26    const filetypes = /jpeg|jpg|png|gif/;
```

The bottom panel shows the TERMINAL output:

```
PS C:\FSD\http-server>
node filesystem.js
Server started on port 3000
```

OUTPUT:



File Upload Server